FIG.1

FIG.2(a)

FIG.2(b)

FIG.2(c)

FIG.2(d)
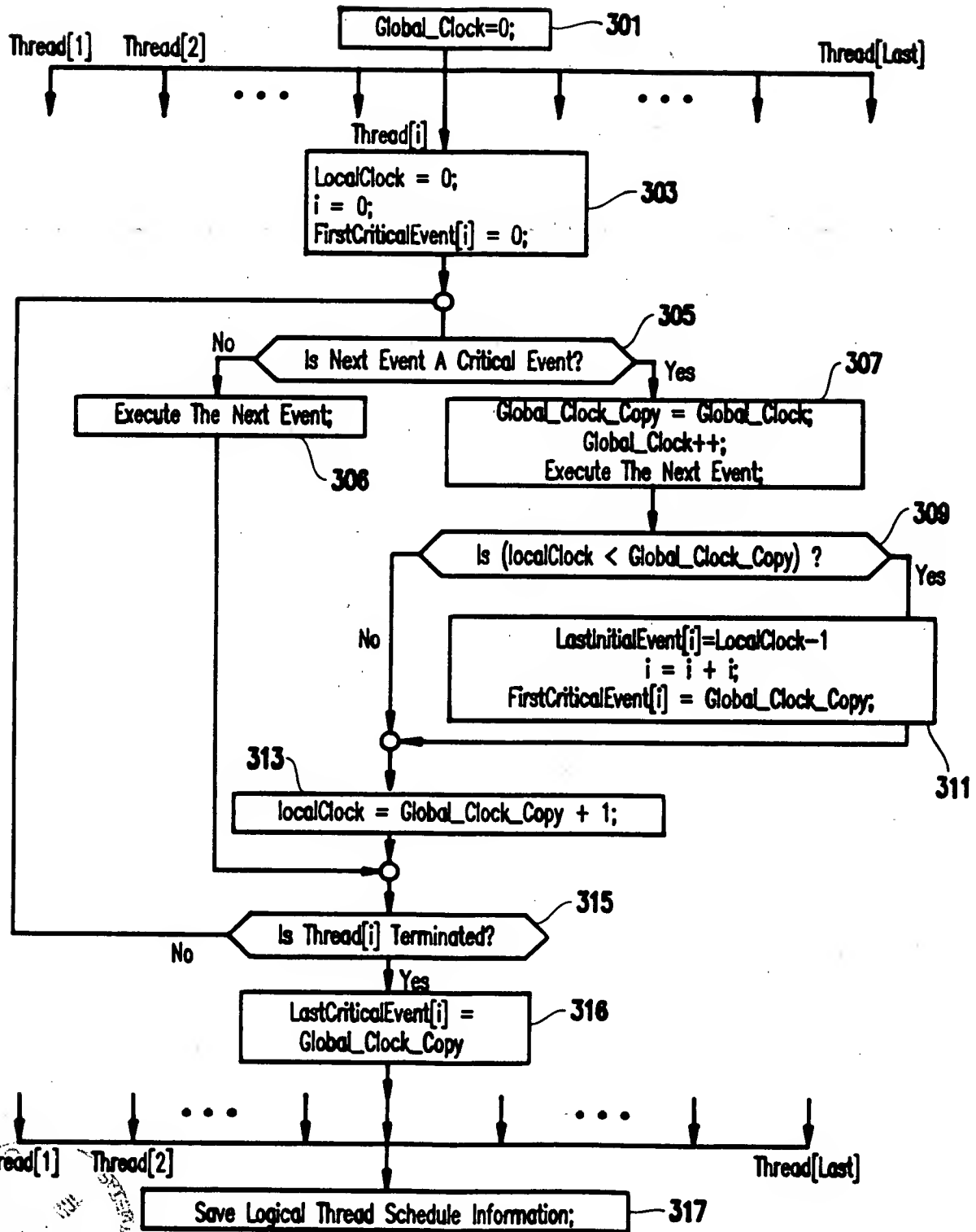
time

Global_Clock=0; — 301

Thread[1]   Thread[2]   Thread[Last]

• • •   • • •

Thread[i]

LocalClock = 0;
i = 0;   303
FirstCriticalEvent[i] = 0;

No   Is Next Event A Critical Event?   — 305   Yes   — 307

Execute The Next Event;   Global_Clock_Copy = Global_Clock;
Global_Clock++;
Execute The Next Event;
308

Is (localClock < Global_Clock_Copy) ?   — 309   Yes

No   LastInitialEvent[i]=LocalClock-1
i = i + i;
FirstCriticalEvent[i] = Global_Clock_Copy;

313   311

localClock = Global_Clock_Copy + 1;

Is Thread[i] Terminated?   — 315

No   Yes

LastCriticalEvent[i] =   — 316
Global_Clock_Copy

↓   ↓   • • •   ↓   ↓   ↓   • • •   ↓

Thread[1]   Thread[2]   Thread[Last]

Save Logical Thread Schedule Information;   — 317

FIG.3(a)

FIG.3(b)

```
                                         301        300
                    ( enterGcCriticalSection )

                                                      302
        ( Global_Counter_Copy = Global_Counter )

                    ( Global_Counter++ )              303

                    ( Execute Critical Event )        304

                    ( leave GcCriticalSection )       305
```

T1   T2   T3   T4

L0   L0   L0   L0

G0

L1   G1

L2   G2

L3   G3

: thread shared variable

: thread local variable

Gi : global clock value

Li : local clock value

L0   G3

L4   G4

L5   G5

Thread Schedule Intervals:

T1 : <0,2>,<7,8>

T3 : <6,6>

T4 : <3,4>

L0   G5

L6   G6

L3   G6

L7   G7

L8   G8

FIG.4

S0: START, i = 0
S1: update FirstCriticalEvent(i) and
     LastCriticalEvent(i)
S2: yield the thread schedule
S3: execute event.
   if CriticalEvent increment global_counter
S4: i = i + 1
S5: END

C1: global_counter < FirstCriticalEvent(i)
C2: not C1
C3: global_counter <= LastCriticalEvent(i)
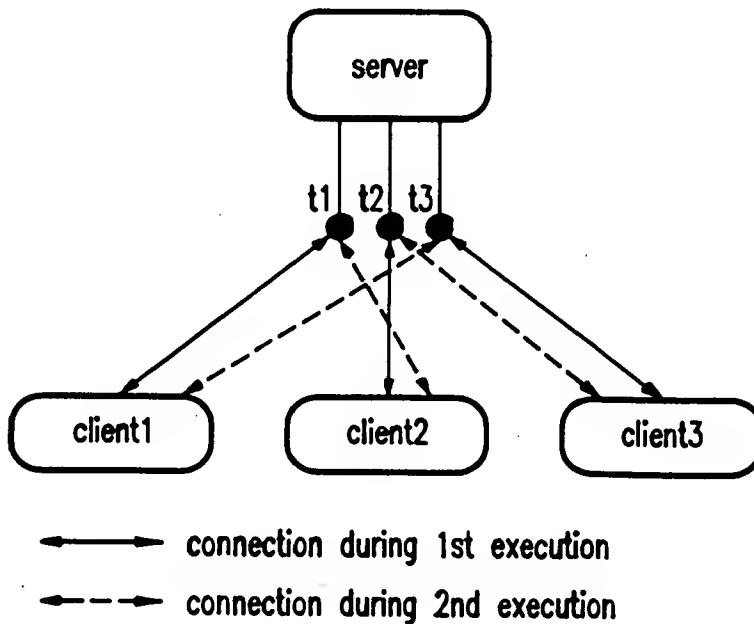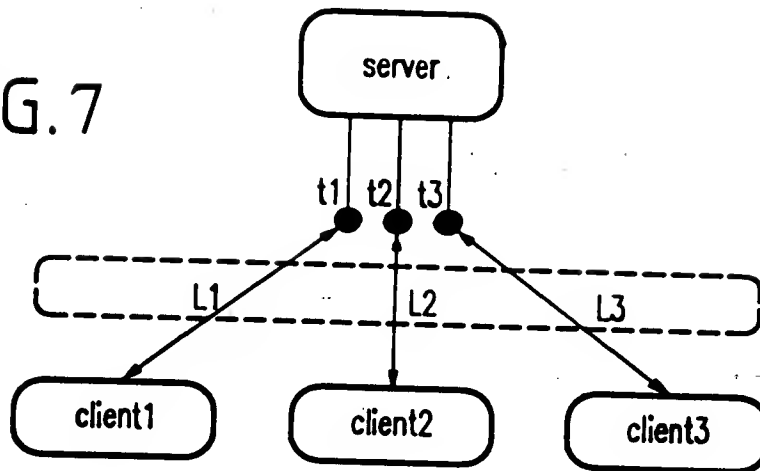C4: not C3
C5: i <= last interval
C6: not C5

FIG.5

connection during 1st execution
connection during 2nd execution

FIG.6

FIG.7

L1: <gS1, Client1Id>,   Client1Id = <Client1VMID, gCounterClient1>>
L2: <gS2, Client2Id>,   Client2Id = <Client2VMID, gCounterClient2>>
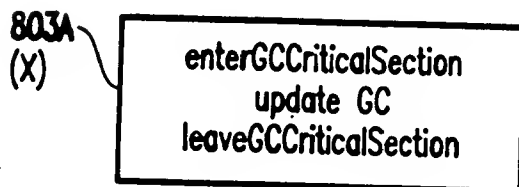L3: <gS3, Client3Id>,   Client3Id = <Client3VMID, gCounterClient3>>
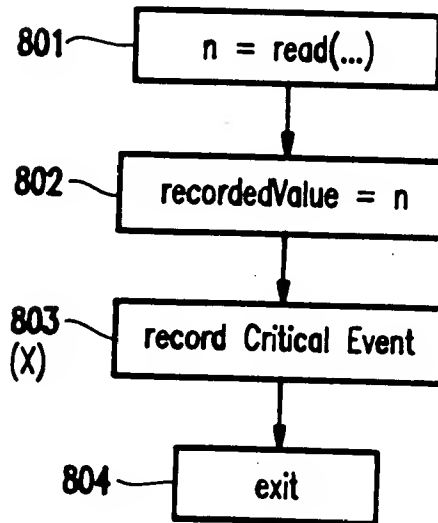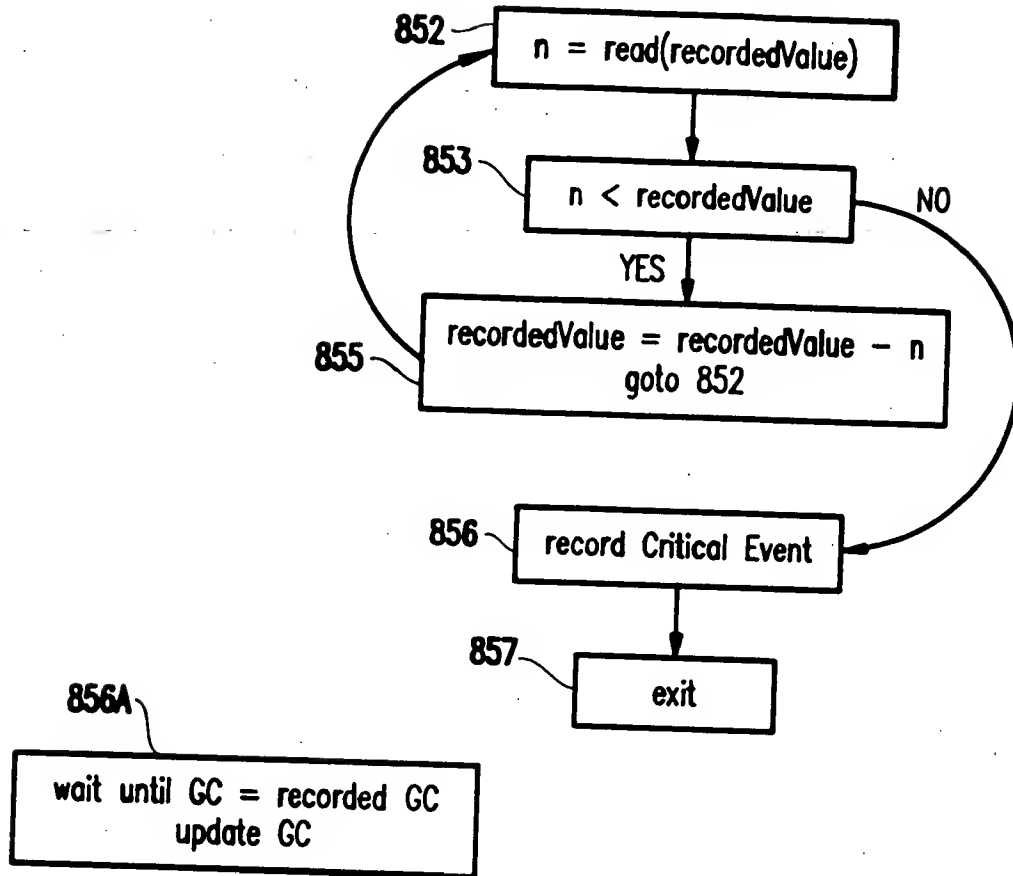
FIG.8(a)

read in Record Mode

FIG.8(b)
read in Replay Mode

852 — n = read(recordedValue)

853 — n < recordedValue — NO

YES

855 — recordedValue = recordedValue − n
goto 852

856 — record Critical Event

857 — exit

856A — wait until GC = recorded GC
update GC

FIG.9(a)
write in Record Mode

901a — enterGCCriticalSection

902a — update GC
write

903a — leaveGCCriticalSection

FIG.9(b)
write in Replay Mode

wait until GC = recorded GC for this write
901b —

902b — write

903b — update GC

FIG.10
accept and connect in Record Mode

Client   1000A

Server   1000B

1001 — send connection request

1002 — accept a connection request

1003 — send ClientId = <clientJVMId, ClientThreadNum> to server

1004 — receive ClientId

1005 — (A) record Critical Event

1006 — log <ServerId, ClientId> into ServerSocketLog, ServerId = networkEventId of accept.

1005A —
(A): enterGCCriticalSection
    update ClientGC
    leaveGCCriticalSection

1007 — (B) record Critical Event

(B): enterGCCriticalSection
    update ServerGC
    leaveGCCriticalSection — 1007B

FIG.12(a)
efficient replay of read

1201 — enterFDCriticalSection(socket)
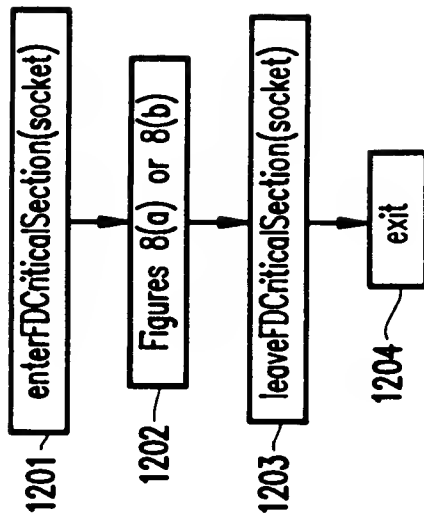
1202 — Figures 8(a) or 8(b)

1203 — leaveFDCriticalSection(socket)

1204 — exit

FIG.12(b)
efficient replay of write

1251 — enterFDCriticalSection(socket)

1252 — Figure 9

1253 — leaveFDCriticalSection(socket)

1254 — exit

1300

FIG.13

FIG.11
accept in Replay Mode
Server

1102 — retrieve ClientId = recValue, from ServerSocketLog for this ServerId, (= networkEventId for this accept)

1103 — check connection pool for recValue. If recValue in connection pool, goto 1107

1004 — accept a connection request (C)

1105 — receive ClientId

1106 — if ClientId notEqual recValue save ClientId in connection pool; goto (C)

1107 — record Critical Event (B)

1108 — exit